PATENTS
15311-2207
200308268-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re The Application of:
Stephen Dickson

Serial No.: 09/428,384

Examiner: Not yet assigned

Filed: October 28, 1999

Art Unit: 2162

For: COMPUTERIZED FILE SYSTEM AND METHOD

Confirmation No.: 4583

Cesari and McKenna, LLP
88 Black Falcon Avenue
Boston, MA 02210
September 14, 2007

CERTIFICATE OF TRANSMISSION

I hereby certify that the following papers are being electronically transmitted to the Patent and Trademark Office by EFS-Web on September 14, 2007

/Michael Reinemann /
Michael R. Reinemann

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

APPEALBRIEF

In response to the Notice of Non-Compliant Appeal Brief mailed August 20, 2007,

Applicant hereby submits this Appeal Brief.

REAL PARTY IN INTEREST

The real party in interest is Hewlett-Packard Development Company, L.P. of Houston, Texas.

RELATED APPEALS AND INTERFERENCES

Applicant and their legal representatives know of no related appeals or interferences that will directly affect, be directly affected by, or have a bearing on the Board's decision in the present appeal.

STATUS OF CLAIMS

Claims 1-40 are pending in the case. Claims 1-40 stand rejected under 35 U.S.C. §§101, 102 and 112, second paragraph, as noted in the non-final Office Action mailed February 21, 2007 that re-opened prosecution. The rejection of claims 1-40 is appealed.

A copy of claims 1-40 is attached hereto as an Appendix.

STATUS OF AMENDMENTS

In response to the non-final Office Action mailed February 21, 2007, Applicant amended claims 19-22 and 24, pursuant to 37 C.F.R. §§1.116 and 41.33(a).

SUMMARY OF CLAIMED SUBJECT MATTER

The summary is set forth in eleven exemplary embodiments that correspond to independent claims 1, 6, 11, 14, 19, 20, 21, 27, 30, 33 and 36. Discussions about elements and recitations of these claims can be found at least at the cited locations in the specification and drawings.

Independent claim 1 is directed to a computerized file system in which a first process (252), which may reside in a server node (202), maintains a data file (250). A second process (260), which may reside at a client node (204), generates a first message (300) that requests

that the second process be granted a plurality of tokens that are required to modify at least one characteristic of the data file. The first process (252) generates a second message (302) that grants the tokens to the second process, assuming they are available. Specification, p. 13, line 1 to p. 15, line 4, and Figs. 3-5.

Independent claim 6 is directed to a computer node (202) including a first process (252) residing in the node that generates a first message (302) that grants a set of tokens to a second process (260) that requested the grant of the set of tokens, where the set of tokens is required for the second process to modify at one characteristic of a file (250). Specification, p. 11, line 12 to p. 15, line 4, and Figs. 2-3.

Independent claim 11 is directed to a computer node (204) including a first process (260) that generates a request to a second process (252) for a set of tokens required to modify at least one characteristic of a file (250). Specification, p. 13, line 1 to p. 15, line 4, and Figs. 3-5.

Independent claim 14 is directed to a network computing system comprising a first node (202) having a data file (250), and a second computer node (204) that issues a first message (300) requesting the grant of a set of tokens for modifying at least one characteristic of the data file, where the first computer node issues a second message (302) to the second computer node that grants the set of tokens if they are available. Specification, p. 11, line 12 to p. 12, line 11, p. 13, line 1 to p. 15, line 4, and Figs. 2-5.

Independent claim 19 is directed to a computer-readable memory (210) containing computer-executable program instructions including a first instruction that maintains a data file (250) in computer storage memory, a second instruction that generates a first message (300) requesting the grant of a plurality of tokens that are required to modify at least one

characteristic of the data file, and a third instruction that generates a second message (302) that grants the tokens, assuming they are available. Specification, p. 11, line 12 to p. 15, line 4, and Figs. 2-5.

Independent claim 20 is directed to a computer-readable memory (210) containing computer-executable program instructions including first instructions that generate a first message (302) granting a set of tokens to a requester (260) of the set of tokens, where the set of tokens is required to modify at least one characteristic of a file (250). Specification, p. 11, line 12 to p. 15, line 4, and Figs. 2-5.

Independent claim 21 is directed to a computer-readable memory (210) containing computer-executable program instructions including first instructions that generate a request (300) for a grant of a set of tokens required to modify by an issuer (260) of the request at least one characteristic of a file (250). Specification, p. 11, line 12 to p. 15, line 4, and Figs. 2-5.

Independent claim 27 is directed to a computerized file system including means for maintaining a data file (processor 212, memory 210, and data file 250 of node 202), means for generating a first message (processor 212 and memory 210 of node 204, and message 300) requesting a grant of a plurality of tokens for modifying at least one characteristic of the file, and means for generating a second message (processor 212 and memory 210 of node 202, and message 302). In response to the first, that grants the tokens if they are available. Specification, p. 11, line 14 to p. 15, line 4, and Figs. 2-5.

Independent claim 30 is directed to a computerized method including maintaining a data file (250), generating a first message (300) requesting a grant of a plurality of tokens to modify at least one characteristic of the file, and generating a second message (302) in

response to the first, that grants the tokens if they are available. Specification, p. 11, line 14 to p. 15, line 4, and Figs. 2-5.

Independent claim 33 is directed to a computerized method including generating a first message (302) that grants a set of tokens that are required to modify at least one characteristic of a file (250). Specification, p. 11, line 14 to p. 15, line 4, and Figs. 2-5.

Independent claim 36 is directed to a computerized method including generating a request (300) for a grant of a set of tokens required to modify at least one characteristic of a file (250). Specification, p. 11, line 14 to p. 15, line 4, and Figs. 2-5.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1-40 are unpatentable under 35 U.S.C. §101 on the grounds that they lack utility, where the claims provide the useful result of reducing the number of messages required to maintain the coherency of computer data files. Whether claims 19-26 are unpatentable under 35 U.S.C. §101 on the grounds that they are non-statutory, software per se, where the claims do not cover abstract ideas, laws of nature or natural phenomenon and, even if they were found to do so, they represent a practical application of an abstract idea or a law of nature.

Whether claims 2-5, 7-10, 12-13, 15-18, 22-26, 28-29, 32, 34-35, and 37 are unpatentable under 35 U.S.C. §112, second paragraph, for using the indefinite article "a" in the preamble of these claims, where the claims employ the well-accepted format for preambles, and clearly recite and distinctly claim the subject matter of Applicant's invention.

Whether claims 1-40 are unpatentable under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 5,634,122 to Loucks et al. (hereafter "Loucks"), where Loucks fails to disclose a single message that either requests or grants multiple tokens.

ARGUMENT

Legal Standard

§101-Utility

An assertion of utility by an applicant creates a presumption of utility that will be sufficient to satisfy the utility requirement of 35 U.S.C. §101. *See, e.g., In re Jolles*, 628 F.2d 1322 (CCPA 1980); *In re Langer*, 503 F.2d 1380 (CCPA 1974). As the Court of Customs and Patent Appeals stated in *In re Langer*:

> As a matter of Patent Office practice, a specification which contains a disclosure of utility which corresponds in scope to the subject matter sought to be patented must be taken as sufficient to satisfy the utility requirement of § 101 for the entire claimed subject matter unless there is a reason for one skilled in the art to question the objective truth of the statement of utility or its scope.

In re Langer, 503 F.2d at 1391 (emphasis in original).

Indeed, to reject claims under 35 U.S.C. §101, the Examiner bears the burden of establishing that it is more likely than not that a person of ordinary skill in the art would not consider that any utility asserted by the Applicant would be specific and substantial. If one skilled in the art can use a claimed invention in a manner which provides some immediate benefit to the public, then the invention is useful. *See, e.g., Raytheon v. Roper*, 724 F.2d 951, 958 (Fed. Cir. 1983), *cert. denied*, 469 U.S. 835 (1984) ("When a properly claimed invention meets at least one stated objective, utility under 35 U.S.C. 101 is clearly shown."). Furthermore, a dependent claim will define an invention that has utility if the independent claim from which it depends defines an invention having utility. *See* M.P.E.P. §2107.02

§112, Second Paragraph

In rejecting claims under 35 U.S.C. §112, second paragraph, the Examiner must show that a person of ordinary skill in the art could not interpret the metes and bounds of the claim

so as to understand how to avoid infringement. *See Morton Int'l, Inc. v. Cardinal Chem. Co.*, 5 F.3d 1464, 1470 (Fed. Cir. 1993); *Metabolite Labs., Inc. v. Lab. Corp. of Am. Holdings*, 370 F.3d 1354, 1366 (Fed. Cir. 2004) ("The requirement to 'distinctly' claim means that the claim must have a meaning discernible to one of ordinary skill in the art when construed according to correct principles …. Only when a claim remains insolubly ambiguous without a discernible meaning after all reasonable attempts at construction must a court declare it indefinite.").

§102

In rejecting claims under 35 U.S.C. §102(e), the examiner bears the burden of showing that each and every element of the claim is found, either expressly or inherently, in a single prior art reference. *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co*., 868 F.2d 1226, 1236 (Fed. Cir. 1989). Furthermore, the elements must be arranged as required by the claim, although the identity of terminology is not required. *In re Bond*, 910 F.2d 831 (Fed. Cir. 1990).

The claims of the present application do not stand or fall together. Instead Applicant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and presented with separate headings and sub-headings as required by 37 C.F.R. §41.37(c)(1)(vii).

Utility

Claims 1-40 stand rejected for failing to meet the utility requirement of 35 U.S.C. §101. Applicant respectfully disagrees, and requests that this rejection be reversed.

To establish a *prima facie* rejection based on lack of utility, the Examiner must

demonstrate that it is more likely than not that a person of ordinary skill in the art would not consider that any utility asserted by the applicant would be specific and substantial, and the Examiner's showing must contain:

> (i) An explanation that clearly sets forth the reasoning used in concluding that the asserted utility for the claimed invention is not both specific and substantial nor well-established;
>
> (ii) Support for factual findings relied upon in reaching this conclusion; and
>
> (iii) An evaluation of all relevant evidence of record, including utilities taught in the closest prior art.

See M.P.E.P. §2107

First, Applicant submits that the Office Action fails to make a *prima facie* showing of a lack of utility. In particular, the Office Action does not contain an explanation clearly setting forth its reasoning. The Office Action simply states that the "bodies of claims 1, 6, 11, 14, 19, 20, 21, 27, 30, 33, and 36 . . . lack[] real world useful result". *See* Office Action at p. 4. Additionally, the Office Action contains no support for any factual findings, nor does it present any relevant evidence or contain any evaluation of such evidence. As the §101 rejection fails to make a *prima facie* showing, it should be reversed. *See* M.P.E.P. §2107.02 ("It is imperative that Office personnel use specificity in setting forth an initial rejection under 35 U.S.C. §101 and support any factual conclusions made in the *prima facie* showing.")

Second, Applicant submits that the claims present a specific and substantial utility, and that the utility of the claimed invention is readily apparent to one of skill in the art. In particular, Applicant's claimed invention provides the useful result of reducing the number of

messages required to maintain the coherency of data files in computer systems, thereby improving efficiency. Applicant directs the Board's attention to the Specification at pages 7-9 and, more specifically to p. 9, which states:

> This permits the number of token request, grant, revocation, and relinquishment messages that need to generated and transmitted according to the present invention to be substantially reduced compared to the prior art.

Additional support for the utility of the present invention may be found in the Specification at pages 13-15 and at Fig. 3, among other places.

Further support for the utility of the claimed invention may also be found in the cited reference, U.S. Pat. No. 5,634,122 to Loucks et al. ("Loucks"), which explicitly recognizes the utility of avoiding conflicting updates to data. *See* Loucks at Col. 1; 38-41, which states as follows:

> Access to files and directories in a file set are synchronized by the file system to avoid conflicting updates to the data and to ensure consistency of data read from the file system.

The utility of the present invention is also reflected in the claims. For example, Claim 1 recites that a second process, in response to a request, receives a message granting it a plurality of tokens for modifying the characteristic of a file, provided that those tokens are available for grant. The consistency of the file is thus ensured, while reducing the number of messages that must be sent. Independent Claim 6 similarly recites the generation of a message granting a set of tokens to a second process that requested them, provided that the tokens are available for grant. Applicant submits that the above-stated utility is reflected in all of the claims of the present invention.

As the claimed invention satisfies the utility requirement of §101, Applicant respectfully requests that the rejection be reversed.

Non-Statutory Software Per Se

Claims 19-26 stand rejected on the grounds that they are "programs or software programs having a plurality of executable instructions or codes that is non-statutory, software per se." *See* Office Action at p. 5. Applicant respectfully disagrees, and request that the rejection be reversed.

Claim 19 recites as follows:

> Computer-readable memory containing computer-executable program instructions, the instructions comprising:
>
> first instructions maintaining a data file in a computer storage memory;
>
> second instructions generating a first message requesting grant of a plurality of tokens required to modify at least one characteristic of said file located in said computer storage memory; and
>
> third instructions generating a second message, in response to said first message, that grants said tokens if said tokens are available for grant to said second process.

First, Applicant submits that claim 19 is not directed to an abstract idea, a law of nature or a natural phenomenon, and thus does not fall within one of the judicial exceptions to §101. As shown above, the first element of claim 16 is directed to instructions that maintain a data file in a computer storage memory. This element does not involve an abstract idea, such as mathematical expression, nor does it involve a law of nature, such as Newton's law gravity, or a natural phenomenon, such as a plant or mineral existing in nature. Similarly, the second and third elements of claim 19 are directed to instructions generating messages that request a grant of tokens and that grant the requested tokens, respectively. Again, such elements are not directed to an abstract idea/mathematical expression, a law of

nature or a natural phenomenon. As claim 19 does not cover a judicial exception to §101, the rejection of claim 19 should be reversed.

Second, even if claim 19 were construed to cover a judicial exception, it represents a "practical application", and is thus patentable subject matter. Specifically, claim 19 produces a useful, concrete and tangible result, namely the issuance of a single message requesting a plurality of tokens required to modify a data file and a single message granting the plurality of tokens, thereby maintaining the coherency of the data file in an efficient manner. As mentioned above, the utility of such a process is recognized in the cited reference. Thus, even if claim 19 were found to cover one of the judicial exceptions to §101, it nonetheless represents a practical application. The rejection of claim 19 should therefore be reversed.

Claims 20-26 depend from claim 19, and thus the rejection of these claims should also be reversed for at least the above-stated reasons.

§112, Second Paragraph

Claims 2-5, 15-16, 18 and 28-29 stand rejected under 35 U.S.C. §112, second paragraph on the grounds that the term "A system" as appearing in the preamble of these claims lack antecedent basis. Claims 7-10 and 12-15 stand rejected under §112, second paragraph on the grounds that the term "A node" as appearing in the preamble of these claims lack antecedent basis. Claims 22-26 stand rejected under §112, second paragraph on the grounds that the term "computer-readable memory" as appearing in the preamble of these claims lack antecedent basis. Claims 32, 34-35 and 37 stand rejected under §112, second paragraph on the grounds that the term "A method" as appearing in the preamble of these claims lack antecedent basis. Applicant respectfully disagrees, and requests that this rejection be reversed.

The use of the indefinite article "a" in the preamble of a claim is well-established. It follows from the requirement that claims adhere to the one sentence requirement, and that claims use an indefinite article when introducing a component for the first time. Applicant submits that claims 2-5, 15-16, 18 and 28-29 clearly and distinctly set forth the subject matter of Applicant's invention, thereby satisfying §112, second paragraph. Accordingly, Applicant requests that the rejection of these claims be reversed.

§102

Claims 1-5

Claim 1 recites as follows:

"A computerized data file system, comprising:"

"a first process that maintains a data file stored in a computer-readable memory", and

"a second process that generates **a first message requesting** that said second process be granted by said first process **a plurality of tokens** required for said second process to modify at least one characteristic of said file stored in said computer-readable memory",

"said first process generating **a second message**, in response to said first message, **that grants said tokens** to said second process if said tokens are available for grant to said second process".

As shown, claim 1 recites, among other things, that there are a **plurality** of tokens being requested by the second process in the **single first message**, and that there are a **plurality** of tokens being granted by the first process in the **single second message**.

Description of the Cited References

Loucks describes a technique for controlling access to shared resources in a distributed computer system using a token manager. *See* Abstract. In order to access the shared resource, a process (e.g., a client process) must hold a single token associated with the

operation. The token represents an authorization for the client process to perform the operation. *See* Abstract, and Col. 6; 8-18. If the client process does not hold the token, it requests the token from a token manager. *See* Col. 6; 12-16. The token manager examines a list of tokens it holds and, depending on the content of the list, may request a new token from a local token manager. Alternatively, it may request that another client process revoke a token, or it may simply grant an available token to the requesting client process. *See* Col. 6; 28-36.

When a token is requested from the local token manager, the local token manager determines if the requested token conflicts with other outstanding tokens. *See* Col. 7; 2-7. If no conflict exists, the local token manager grants the token. Otherwise, if a conflict exists (e.g., the token is granted to another process), the local token manager first revokes the token from the process holding the token, and then grants the token to the requesting client process. *See* Col. 7; 6-12. Specific token requests may cause the revocation of multiple tokens. For example, a request for an exclusive write token on a file causes the token manager to revoke all granted read tokens on that file before granting the exclusive write token to the requesting process. *See* Col. 7; 9-12.

In summary, with Loucks, a process requests a single token from a token manager in order to access a shared resource. The token manager, in turn, either grants the token immediately, requests the token from a local token manager, or revokes the token from another process.

In rejecting claim 1, the Examiner cites the following three excerpts of Loucks as purportedly teaching a single message that requests a plurality of tokens in order to modify at least one characteristic of a file:

1) Col. 3; 45-50;

2) Col. 6; 8-35; and

3) Col. 8; 35-67.

Applicant submits that none of these excerpts from Loucks discloses a single message that requests a plurality of tokens.

The excerpt from Col. 3; 45-50 states as follows:

> means for requesting authorization for a computer implemented process to perform an operation on shared data; means for managing requests generated by the means for requesting; and means for granting authorization tokens allowing an operation to be performed on the shared data.

Applicant agrees that this excerpt states that requests for authorization are made, and that multiple tokens are granted. However, this excerpt fails to disclose a single message that either requests multiple tokens, or a single message that grants multiple tokens. Instead, the cited except merely states that authorization is requested, and that authorization tokens are granted. Applicant submits that it is improper to read this excerpt from Loucks as disclosing a single message that requests a plurality of tokens or a single message that grants a plurality of tokens.

In fact, a careful review of Loucks reveals that Loucks actually discloses that tokens are requested and granted one at a time, i.e., one token per message. In particular, the following excerpts from Loucks all show that its tokens are requested one at a time:

(1) Col. 6; 28-29 ("When the token requester, mtkr 418, of an MDFS client requests a token from the token manager . . .");

(2) Col. 6; 31-32 ("it may request a new token from the local token manager . . .");

(3) Col. 7; 4-5 ("If the requested token does not conflict . . .");

(4) Col. 9; 36-37 ("if the Stash token is requested . . .");

(5) Fig. 8A, block 804 ("Request token from ltkm");

(6) Fig. 8B, block 816 ("Wait for token request");

(7) Fig. 9A, block 903 ("Get Open_Write token for fileset y");

(8) Fig. 9B, block 938 ("Identify Requester and requested token")

As shown, what Loucks actually discloses is that tokens are requested one at a time.

The other excerpts from Loucks similarly fail to disclose a single message that requests a plurality of tokens. More specifically, at Col. 6; 8-35, Loucks states as follows:

> mtkr (418) is a component of MCM called the "mobile token requester". Tokens represent an authorization for a process to perform a certain function, e.g. a "read" token permits a client to read data while a "write" token permits the client to update data. The function of token requester 418 is to request tokens, when needed, from the MDFS PX token manager's component mtkm 420 explained below. mtkr requests these tokens on behalf of its local users. It also maintains a list of the acquired tokens. When it receives a "token revoke" request from mtkm, it returns the required token to mtkm.
>
> mtkm (420) is the mobile token manager for MDFS. The function of mtkm is three fold:
>
> 1. Request tokens from the local token manager, lktm 420 (explained below), on behalf of the MDFS clients,
> 2. Maintain a list of all the tokens acquired for its client.
> 3. Send "Revoke token" requests to clients when the specific token is requested by another MDFS client or by the local token manager.
>
> When the token requester, mtkr 418, of an MDFS client requests a token from the token manager 420, the token manager examines the list of tokens it holds on behalf of all its clients. Depending on the contents of this list, it may request a new token from the local token manager, request "revoke token" from one or more clients, or simply grant an available token to the requesting client. [The token types of the Mobile Distributed File System are illustrated in FIG. 7.]

As with the above excerpt, Applicant acknowledges that this excerpt states that Loucks' mobile token requester requests "tokens". However, this excerpt does not disclose the generation of a single message that requests a plurality of tokens, and a fair and proper reading of Loucks limits its disclosure to a single token per message.

The third excerpt of Loucks, at Col. 8; 35-67, states as follows:

> Two types of tokens are defined to support synchronization in these file systems:
>
> --Open Tokens
>
> --Lock Tokens
>
> Open Tokens: These are requested and granted for a volume (fileset) or for individual files in the fileset. There are three possible open tokens:
>
> Open.sub.-- Read,
>
> Open.sub.-- Write,
>
> Open.sub.-- Exclusive
>
> The Open.sub.-- Read 602 and Open.sub.-- Write 602 are shared tokens, they can be issued to multiple clients of a token manager at the same time. The Open.sub.-- Exclusive token can be given to only one client at any time, and it conflicts with other open tokens. The compatibility matrix 600 in FIG. 6a represents the compatibility relations between the open tokens.
>
> A Protocol Exporter PX for a distributed file system requests Open.sub.-- Read or Open.sub.-- Write token for a file set when it "exports" the file set for read or wrote. A distributed file system client cache manager requests Open.sub.-- Read or Open.sub.-- Write token when it mounts (or imports) the fileset. The Open.sub.-- Exclusive token is usually requested only by tools such as a repair or disk check utility. A PX for a distributed file system must acquire the appropriate fileset open token before it is granted any open tokens for individual files in the fileset. When a PX acquires an Open.sub.-- Read token for a fileset, it can only acquire Open.sub.-- Read token for files in the file set. When a PX acquires an Open.sub.-- Write token or an Open.sub.-- Exclusive for a fileset, it can acquire any Open token for individual files in the file set. This scenario also applies for cache managers of distributed file system clients.

Again, Applicants acknowledge that Loucks describes different types of tokens that can be requested and granted. Nonetheless, there is no disclosure by Loucks that a process can generate a single message requesting a plurality of tokens.

In sum, there is no disclosure by Loucks of a process that generates a single message requesting a plurality of tokens. Accordingly, Loucks fails to anticipate claim 1, and the rejection of claim 1 should therefore be reversed.

The rejection of claims 2-5, which depend from claim 1, should also be reversed for at least these reasons.

Claims 6-10

Independent claim 6 recites in relevant part:

"a first message that grants a set of tokens . . . the set of tokens being required for the second process to be able to modify at least one characteristic of a file".

The Office Action cites to Col. 9; 18-30 and 40-48, and Col. 11; 18-32 of Loucks as purportedly anticipating claim 6. Applicant respectfully disagrees.

The cited excerpts from Col. 9 of Loucks state as follows:

In addition to the Open tokens 702 and Lock tokens 704, the present invention introduces two new token types to support the above clients: Stash token 706 and Re-integration token 708.

The Stash token is acquired for an entire fileset or for individual files. The stash token is a read-only shared token. It is requested by an MDFS client cache manager (MCM) to indicate this client's intention to disconnect from the server. The client MCM requests the Stash token from the Mobile Protocol Exporter MPX [See FIG. 5] for an entire fileset. MCM does not need to request stash tokens for the individual files in the fileset. MPX requests the Stash token on the fileset from the local token manager ltkm.

* * *

Since requesting a Stash token indicates that the client is getting ready to disconnect and is probably in desperate need for files in that fileset, an option "as is" is defined to allow the client to get the files. The client can request the Stash token with the "as is" option. If an Open_Exclusive token for the fileset is being held by another client or by a local process on the server, it usually means that there is some repair operation on the fileset. The requesting client will get the stash token with a return code that indicates the fileset structure may be changing. Thus, a request for a Stash token with the "as is" option is always granted.

The cited excerpt from Col. 11 of Loucks states as follows:

If mtkr has not previously acquired the requested token (N output of 906), it sends an Open_Write token request to the MDFS server token manager mtkm (922) and waits for reply (924). If the reply was to grant the token (Y output of 926), the mtkr proceeds to grant the token (916) as described above. If the reply from mtkm was to deny the token request or if time-out was reached before a reply is received (N output of 926), then mtkr proceeds to deny the token request (918) as described above. In both cases, this ends the processing of the current token request operation at mtkr.

> FIG. 9b represents the steps executed by the mtkm. They are similar to those explained in description of FIG. 8, with the "Open_Write token" request for fileset y replaced by "Open_Write token" request for file f.

As indicated above, Applicant acknowledges that Loucks describes multiple types of tokens. However, the issue is whether Loucks discloses a process that generates a single message granting a plurality of tokens required to modify a file. Applicant respectfully submits that Loucks fails to provide such a disclosure.

First, the above excerpts of Loucks do not disclose the granting of a plurality of tokens with a single message. Instead, the above excerpts from Col. 9 disclose the use of a different type of token depending on the particular circumstances, including a Stash token that can be used to obtain a fileset even though the fileset may be undergoing a change. The above excerpt from Col. 11 discloses a request for a single token (the Open_Write token), which either results in the single Open_Write token being granted or in the request being denied. In sum, the cited excerpts fail to disclose a process that generates a single message granting a plurality of tokens for modifying a file.

A review of other portions of Loucks further demonstrates that a grant is only made of a single token, not a plurality of tokens as recited in claim 6. At Col. 6; 28-34, for example, Loucks states as follows:

> When a token requester, mtkr 418, of an MDFS client requests a token from the token manager 420, the token manager examines the list of tokens it holds on behalf of all its clients. Depending on the contents of this list, it may request a new token from the local token manager, request "revoke token" from one or more clients, **or simply grant an available token to the requesting client**.

(emphasis added).

At Col. 10; 24-27, Loucks states:

> If an Open_Exclusive token has been previously granted to mtkm (Y output for 824), or if no such token has been previously granted (N output for 822), the ltkm **grants the requested token** (826).

(emphasis added).

At lines 27-39 of Col. 10, Loucks states:

> When a reply is received that indicates the token is relinquished by its current holder (N out of 844), ltkm proceeds **to grant the token** (826).

(emphasis added).

Finally, at Col. 11, lines 13-18, Loucks states:

> If there is no conflict (N output of 908), or if the process holding the conflicting token ends its file system call (Y output of 912), then mtkr proceeds **to grant the requested token** (916). **Granting a token** involves updates to mtkr data structures and sending a "grant token request" to the requesting process.

(emphasis added).

All of the above excerpts demonstrate that Loucks discloses the grant of a single token. There is no disclosure by Loucks for a single message that somehow grants a plurality of tokens. Because Loucks fails to disclose a single message for granting a plurality of tokens, the rejection of claim 6 should be reversed.

The rejection of claims 7-10, which depend from claim 6, should also be reversed for at least these reasons.

Claims 11-40

Independent claims 11, 14, 19, 21, 27, 30, and 36 likewise recite a first message that requests "a set" or "a plurality" of tokens. As set forth above, Loucks fails to disclose such a message. Accordingly, the rejection of these claims and the claims that depend from them should also be reversed.

Independent claims 20 and 33 likewise recite "a first message that grants a set of tokens". As described above in connection with independent claim 6, Loucks fails to disclose such a message. Therefore, the rejection of these claims and the claims that depend from them should also be reversed.

In sum, Applicant submits that Loucks fails to anticipate the claimed invention, and that the rejection of claim 1-40 therefore should be reversed.

CONCLUSION

Applicant respectfully submits that the claims are allowable over the art of record.

Accordingly, Applicant request that the rejection of all claims be reversed.

Respectfully submitted,

/Michael R. Reinemann/
Michael R. Reinemann
Reg. No. 38,280
Tel. 617-951-3060

Please direct all correspondence to:

IP Administration Legal Department,
M/S 35
Hewlett-Packard Co.
P.O. Box 272400
Fort Collins, CO 80527-2400

CLAIMS APPENDIX
(Claims on Appeal in Appl. Ser. No. 09/428,384)

1. A computerized data file system, comprising:

a first process that maintains a data file stored in a computer-readable memory; and

a second process that generates a first message requesting that said second process be granted by said first process a plurality of tokens required for said second process to modify at least one characteristic of said file stored in said computer-readable memory;

said first process generating a second message, in response to said first message, that grants said tokens to said second process if said tokens are available for grant to said second process.

2. A system according to claim 1, wherein:

said first process is resident at a server computer node, and said second process is resident at a client computer node.

3. A system according to claim 1, wherein:

if any of said tokens are unavailable for grant to said second process as a result of current grant of said tokens to at least one other process, said first process generates a third message revoking the current grant of said tokens to said at least one other process.

4. A system according to claim 3, wherein:

said at least one other process, in response to said third message, generates a fourth message making said tokens available for grant by said first process.

5. A system according to claim 3, wherein:

said first process resides in a first computer node;

said second process resides in a second computer node;

said at least one other process resides in at least one other computer node; and

said first computer, second computer, and at least one other computer nodes are networked together and are remote from each other.

6. A computer node, comprising:

a first process residing in said node that generates a first message that grants a set of tokens, if the set of tokens is available for grant, to a second process that requested grant of the set of tokens, the set of tokens being required for the second process to be able to modify at least one characteristic of a file stored in a computer-readable memory within the computer node.

7. A node according to claim 6, wherein:

the second process resides in a remote computer node.

8. A node according to claim 7, wherein:

one of the first and second processes resides in a server computer node and the other of the processes resides in a client computer node.

9. A node according to claim 6, wherein:

if at least one token in the set of tokens is unavailable for grant because the at least one token is currently granted to a third process, the first process also generates a second message that revokes current grant of the at least one token to the third process prior to generating the first message.

10. A node according to claim 6, wherein:

the first message is generated by the first process in response to a request for the grant of the set of tokens generated by the second process, the request specifying all tokens required for the second process to be able to modify the at least one characteristic of the file.

11. A computer node, comprising:

a first process residing in said node that generates a request to a second process for grant of a set of tokens required to enable the first process to modify at least one characteristic of a file residing in a remote computer-readable memory.

12. A node according to claim 11, wherein:

the second process resides in a second computer node, and the memory is comprised in said second node.

13. A node according to claim 11, wherein:

the set of tokens comprises all tokens required for the first process to be able to modify the at least one characteristic of the file.

14. A network computer system, comprising:

a first computer node having a data file stored in a computer-readable memory; and

a second computer node that issues to the first computer node a first message requesting grant of a set of tokens required to carry out a modification of at least one characteristic of said file stored in the first computer node;

the first computer node issuing a second message to the second computer node after receipt of the first message, the second message granting the set of tokens to the first process if the set of tokens is available for grant to the second process.

15. A system according to claim 14, wherein:

the first computer node is a server node, and the second computer node is a non-server node.

16. A system according to claim 14, wherein:

the set of tokens comprises all tokens required to carry out the modification of the at least one characteristic of the file.

17. A system according to claim 14, wherein:

if at least one token in the set of tokens is unavailable for the grant because the at least one token is currently granted, the first computer node waits to issue the first message

until after the first computer node receives a third message from a third computer node indicating relinquishment of current grant of the at least one token.

18. A system according to claim 17, wherein:

the at least one token comprises a plurality of tokens.

19. Computer-readable memory containing computer-executable program instructions, the instructions comprising:

first instructions maintaining a data file in a computer storage memory;

second instructions generating a first message requesting grant of a plurality of tokens required to modify at least one characteristic of said file located in said computer storage memory; and

third instructions generating a second message, in response to said first message, that grants said tokens if said tokens are available for grant to said second process.

20. Computer-readable memory containing computer-executable program instructions, the instructions comprising:

first instructions generating a first message that grants a set of tokens, if the set of tokens is available for grant, to a requester of the set of tokens, the set of tokens being required to permit the requester to be able to modify at least one characteristic of a file stored in computer storage memory.

21. Computer-readable memory containing computer-executable program instructions, the instructions comprising:

first instructions generating a request for grant of a set of tokens required to enable modification by an issuer of the request of at least one characteristic of a file residing in storage memory.

22. Computer-readable memory according to Claim 19, further comprising:

further instructions generating a third message, if any of said tokens are unavailable for grant as a result of a current grant of said tokens, revoking the current grant of said tokens.

23. A computer-readable memory according to claim 22, wherein:

said further instructions, in response to said third message, generate a fourth message making said tokens available for grant.

24. Computer-readable memory according to claim 20, further comprising:

further instructions generating a second message, if at least one token in the set of tokens is unavailable for grant because the at least one token is currently granted, that revokes previous grant of the at least one token prior to generating the first message.

25. Computer-readable memory according to claim 20, wherein:

the first message is generated in response to a request for the grant of the set of tokens generated, the request specifying all tokens required to be able to modify the at least one characteristic of the file.

26. Computer-readable memory according to claim 21, wherein:

the set of tokens comprises all tokens required to be able to modify the at least one characteristic of the file.

27. A computerized data file system, comprising:

means for maintaining a data file stored in a computer-readable memory; and

means for generating a first message requesting grant of a plurality of tokens required to modify at least one characteristic of said file stored in said computer-readable memory;

means for generating a second message, in response to said first message, that grants said tokens if said tokens are available for grant.

28. A system according to claim 27, further comprising:

means for generating, if any of said tokens are unavailable for grant as a result of current grant of said tokens, a third message revoking the current grant of said tokens.

29. A system according to claim 28, further comprising:

means for generating, in response to said third message, a fourth message making said tokens available for grant.

30. A computerized method for coherently maintaining and modifying a data file, comprising:

maintaining the data file in a computer-readable memory;

generating a first message requesting grant of a plurality of tokens required to modify at least one characteristic of said file in said computer-readable memory; and

generating a second message, in response to said first message, that grants said tokens if said tokens are available for grant.

31. A method according to claim 30, further comprising:

if any of said tokens are unavailable for grant as a result of current grant of said tokens to at least one other process, generating a third message revoking the grant of said tokens.

32. A method according to claim 31, wherein:

in response to said third message, a fourth message making said tokens available for grant is generated.

33. A computerized method for use in maintaining coherency of a data file stored in a computer-readable memory, comprising:

generating a first message that grants a set of tokens, if the set of tokens is available for grant, to a requester of the grant of the set of tokens, the set of tokens being required for requester to be able to modify at least one characteristic of the file stored in the computer-readable memory.

34. A method according to claim 33, wherein:

if at least one token in the set of tokens is unavailable for grant because the at least one token has been currently granted, the method also comprises a second message that revokes current grant of the at least one token prior to generating the first message.

35. A method according to claim 33, wherein:

the first message is generated in response to a request for the grant of the set of tokens generated by the requester, the request specifying all tokens required for the requester to be able to modify the at least one characteristic of the file.

36. A computerized method for use in maintaining coherency of a data file stored in a computer-readable memory, comprising:

generating a request for grant of a set of tokens required to enable modification of at least one characteristic of the file stored in the computer-readable memory.

37. A method according to claim 36, wherein:

the set of tokens comprises all tokens required to be able to modify the at least one characteristic of the file.

38. The system according to claim 1, wherein:

said second process, in response to receiving said second message, modifies said at least one characteristic of said file stored in said computer-readable memory.

39. The system according to claim 27, further comprising:

means for modifying said at least one characteristic of said file stored in said computer-readable memory.

40. The method according to claim 30, further comprising:

modifying said at least one characteristic of said file in said computer-readable memory.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.